# Applying WinWin to Quality Requirements: A Case Study

**Hoh In**
Texas A&M University
Computer Science Department
College Station, TX 77845-3112
+1 979 458-1547
hohin@cs.tamu.edu

**Barry Boehm**
University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
+1 213 740-8163
boehm@sunset.usc.edu

**Thomas Rodgers, Michael Deutsch**
Texas A&M University
Graduate School of Business
College Station, TX 77845-4217
+1 979 845-3139
{trodgers, mdeutsch}@cgsb.tamu.edu

**ABSTRACT**
This paper describes the application of the WinWin paradigm to identify and resolve conflicts in a series of real-client, student-developer digital library projects. The paper is based on a case study of the statistical analysis of 15 projects and an in-depth analysis of one representative project. These analyses focus on the conflict resolution process, stakeholders' roles and their relationships to quality artifacts, and tool effectiveness. We show that stakeholders tend to accept satisfactory rather than optimal resolutions. Users and customers are more proactive in stating win conditions, whereas developers are more active in working toward resolutions. Further, we suggest that knowledge-based automated aids have potential to significantly enhance process effectiveness and efficiency. Finally, we conclude that such processes and tools have theoretical and practical implications in the quest for better software requirements elicitation.

**Keywords**
Requirements Engineering, Software Quality Attributes, Risk, Conflict Identification and Resolution, Software Cost Analysis.

## 1. INTRODUCTION

Negotiating stakeholder win-win relationships among software quality requirements is a technique that emerged during the 1990's in order to overcome the difficulties arising from contract-oriented specification compliance (popular in the 1970's) and service-oriented customer satisfaction (popular in the 1980's). Quality attributes (e.g., dependability and evolvability) of software intensive systems and their attendant conflicts are frequently unexplored which eventually results in stakeholder disappointment. Obstacles to adoption of negotiated win-win relationships for quality attributes include coordination of multiple stakeholder interests and priorities, reasoning of complicated dependencies, and scalability of an exponentially increasing resolution option space. This case study of conflict identification and resolution techniques was performed because:

- Identifying quality requirements and resolving conflicts is difficult.
- Studying such a process provides theoretical and practical lessons.

Based upon prior research, we start with an assumption that the WinWin negotiation model is effective in surfacing and resolving system quality attribute conflicts between stakeholders. The following two research questions are addressed in this paper.

1. **Can behavioral differences between stakeholders be observed within this application of the WinWin negotiation model?**
2. **Do knowledge-based automated aids enhance surfacing and resolving system quality attribute conflicts?**

The following sections address: approaches to surfacing and resolving quality attribute conflicts; a simple case to illustrate this process; empirical findings from a series of projects; future directions for research and practice; related work; and conclusion.

## 2. THE APPROACH

An assumption is made that process and tools can surface quality attribute conflicts. Two approaches to conflict resolution are (1) a manual approach based on the WinWin model and (2) a semi-automated approach that incorporates knowledge-based tools.

### Challenges

In order to explicitly surface such conflicts, obstacles must be recognized. The process is inherently complex and difficult for many reasons including the following:

- *Stakeholder interests and priorities are different.* Users feel that full functionality, dependability, and ease of use are the most important attributes. Customers are concerned about cost and schedule. Developers are primarily concerned with low project risk and reusing assets. Maintainers are strongly concerned with good



0-7695-1050-7/01 $10.00 © 2001 IEEE

diagnostics and easy maintenance. Finding the middle ground among these requirements commitments is difficult.

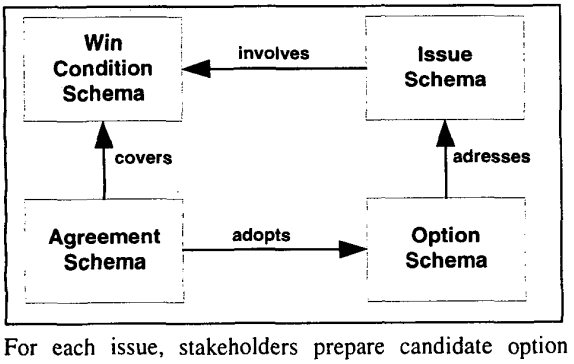
- *Dependencies among quality attributes are complex.* Every decision to improve a set of quality attributes may impact others, particularly cost and schedule. Some requirement decisions may not be compatible with others.
- *Conflict resolutions* exist *within an exponentially increasing solution space.* For example, a cost overrun conflict requires resolution of the following types of questions related to quality attributes. Which modules should be reduced and by how much to get the project back on track? Which modules can be degraded in terms of their quality attributes? How much of which qualities should be degraded?

**Manual Approach to Surfacing and Resolving Quality Attribute Conflicts**

A manual approach can be based on the WinWin Spiral Model [1, 2]. The WinWin model provides a general framework for identifying and resolving requirement conflicts by drafting and negotiating artifacts such as win conditions, issues, options, and agreements. The WinWin model uses Theory W, "Make everyone a winner", [7] to generate the stakeholder win-win situation incrementally through the Spiral Model. WinWin assists stakeholders to identify and negotiate issues (i.e., conflicts among their win conditions), since the goal of Theory W involves stakeholders identifying their win conditions, and reconciling conflicts among win conditions.

Figure 1 shows the negotiation model used by WinWin, in terms of its primary schemas and the static relationships between them. Stakeholders begin by entering their win conditions, using a schema provided by the WinWin model. If a conflict among stakeholders' win conditions is determined, an issue schema is composed, summarizing the conflict and the win conditions it involves.

*Figure 1. The WinWin Negotiation Model*




For each issue, stakeholders prepare candidate option schemas addressing the issue. Stakeholders then evaluate the options, iterate some, agree to reject others, and ultimately converge on a mutually satisfactory (i.e., win-win) option. The adoption of this option is formally proposed and ratified by an agreement schema, including a check to ensure that the stakeholders' iterated win conditions are indeed covered by the agreement. Usage experience also indicates that WinWin is not a panacea for all conflict situations, but generally increases stakeholders' levels of cooperation and trust [8, 9].

**Semi-automatic Approach to Surfacing and Resolving Quality Attribute Conflicts**

Our experience with WinWin usage indicates that, as applications reach the size of several dozen win conditions, it becomes hard for stakeholders to manually identify likely conflicts among them. Thus, we have been experimenting with automated aids such as QARCC (Quality Attribute Risk and Conflict Consultant) and S-COST (Software Cost Option Strategy Tool) to help stakeholders identify issues and formulate options for resolving them.

QARCC [4] is an exploratory knowledge-based tool for identifying potential conflicts and risks among quality requirements early in the software life cycle. QARCC helps stakeholders identify potential quality issues and draft issues in WinWin. QARCC uses the "Attributes" portion of WinWin's domain taxonomy to identify potential quality attributes conflicts. As stakeholders enter win conditions, they identify which domain taxonomy elements are relevant.

The seven top-level quality attributes in the WinWin domain taxonomy are (1) Dependability, (2) Interoperability, (3) Usability, (4) Performance, (5) Evolvability & Portability. (6) Cost & Schedule, and (7) Reusability. Suppose a stakeholder enters a win condition schema and puts "Performance" in the Taxonomy Elements slot. QARCC will then draw on its knowledge base to analyze potential conflicts between Performance and other quality attributes. It will then notify the affected stakeholders of the potential conflicts.

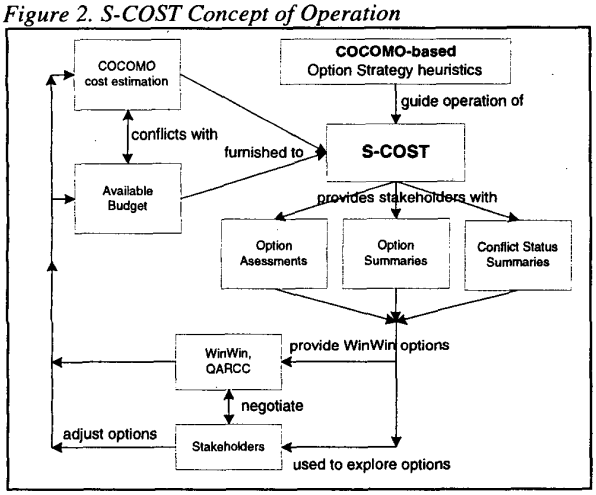
QARCC also recognizes six types of stakeholders: (1) General Public, (2) Interoperator, (3) User, (4) Maintainer, (5) Developer, and (6) Customer. To determine which stakeholders to notify of a potential conflict between Performance and Cost & Schedule, QARCC uses the Stakeholder/Quality-Attribute relationships shown later in Figure 5. In this case, QARCC would notify the User and Customer stakeholders of the potential conflict, as they are the stakeholders generally most concerned with Performance.

QARCC is good for making top-level suggestions about potential quality attribute conflicts, but it lacks detail. S-COST (Software Cost Option Strategy Tool) [5, 6] is an effort to provide such detail in the area of quality attribute conflicts involving cost. S-COST helps stakeholders draft

options for potential cost conflict resolution and facilitate their negotiation to reach an agreement. S-COST extends WinWin and complements QARCC by using an additional software cost knowledge base related to another component of WinWin, COCOMO (COnstructive COst MOdel) [3], which helps identify cost conflicts with functional requirements. S-COST uses the COCOMO cost drivers, cost estimates, and related experience base to sharpen QARCC's identification of cost-conflict issues. It also suggests in- depth cost-conflict resolution options, and provides option visualization and negotiation aids to help stakeholders resolve the cost-conflict issues.

The S-COST concept of operation is shown in Figure 2. Using the WinWin model, stakeholders enter their new win conditions. These may involve functions, quality goals or constraints. As shown in Screen 1 of Figure 3, win condition schemas have attributes such as Priority and domain Taxonomy Elements. For win conditions with quality attribute and cost/ schedule Taxonomy Elements, QARCC examines its Quality Attribute Strategies [6] to search for potential conflicts. For example, layering the architecture to meet the Portability win condition in Screen 1 produces likely conflicts with Cost/Schedule and Performance (Screen 2. In the initial version of QARCC, Cost and Schedule were combined into Development Affordability, and Performance was called Efficiency). QARCC then generates a draft issue identifying this potential conflict for stakeholders to consider (Screen 3).

*Figure 2. S-COST Concept of Operation*




Continuing with the scenario in Figure 2, once stakeholders are presented with a set of draft issues from QARCC involving cost, they can then use COCOMO to analyze the potential cost conflicts identified. If the resulting estimated cost exceeds the target cost, a stakeholder enters this as a WinWin issue whose solution needs to be negotiated by the stakeholders. As indicated in Figure 2, S-COST operates on the issue and COCOMO estimate information to:

- Suggest options for resolving cost issues (screen 4 in Figure 3) *(Option Assessments)*;
- Apprise affected stakeholders of their availability and implications *(Option Summaries)*;
- Visualize other stakeholders' options and facilitate negotiation for resolving cost conflicts. *(Option Status Summaries)*

After the stakeholders converge on a mutually satisfactory (win-win) combination of options, they draft an agreement schema, and follow WinWin's procedures for voting and adopting the agreement.

## 3. DIGITAL LIBRARY PROJECT CASE STUDY

The WinWin model was used to identify and resolve conflicts among software requirements for 15 digital library systems. These digital library systems were developed with input from the USC library staff (as customers and users) and graduate students (as developers) enrolled in CSCI 577A, Software Engineering, during the 1996-7 school year [8, 9]. The library projects dealt with diverse data, including medieval manuscripts, technical reports, corporate business information, and stereoscopic slides. Each project had its desired capabilities, and unique constraints. The 86 graduate students in the class, broken up into 15 teams (5 or 6 students per team), formulated operational concepts, requirements specifications, architectures, prototypes, life cycle plans, and integrating rationale for the proposed capabilities of 15 USC Library Information Systems during the 11-week semester. Altogether, 12 project topics were chosen for this experiment, six of which were continued during the following semester. One team could not be evaluated due to missing electronic data.

Team 2's project (Latin American Pamphlets) is used as an example. The customer defined problem statement follows:

> *The purpose of this project is to create an attractive user-friendly prototype for a virtual archive (i.e., a virtual framework for virtual items or collection groups within a larger collection) of research materials primarily in Spanish, with examples of other languages donated by Mr. and Mrs. Herbert F. Boeckmann II. A challenge for the student team could be to develop a cost effective and timely way to organize and make the entire collection widely accessible not only locally but globally.*

Team 2 represents an average team in terms of grade and experience. The team was highly productive and as a group considered itself to have a very good understanding of software considerations along with increasing confidence, trust and willingness to "do it again." The team was able to identify and resolve quality conflicts within a three day time period.

*Figure 3. Sample Screens from WinWIn, QARCC, and S-COST*

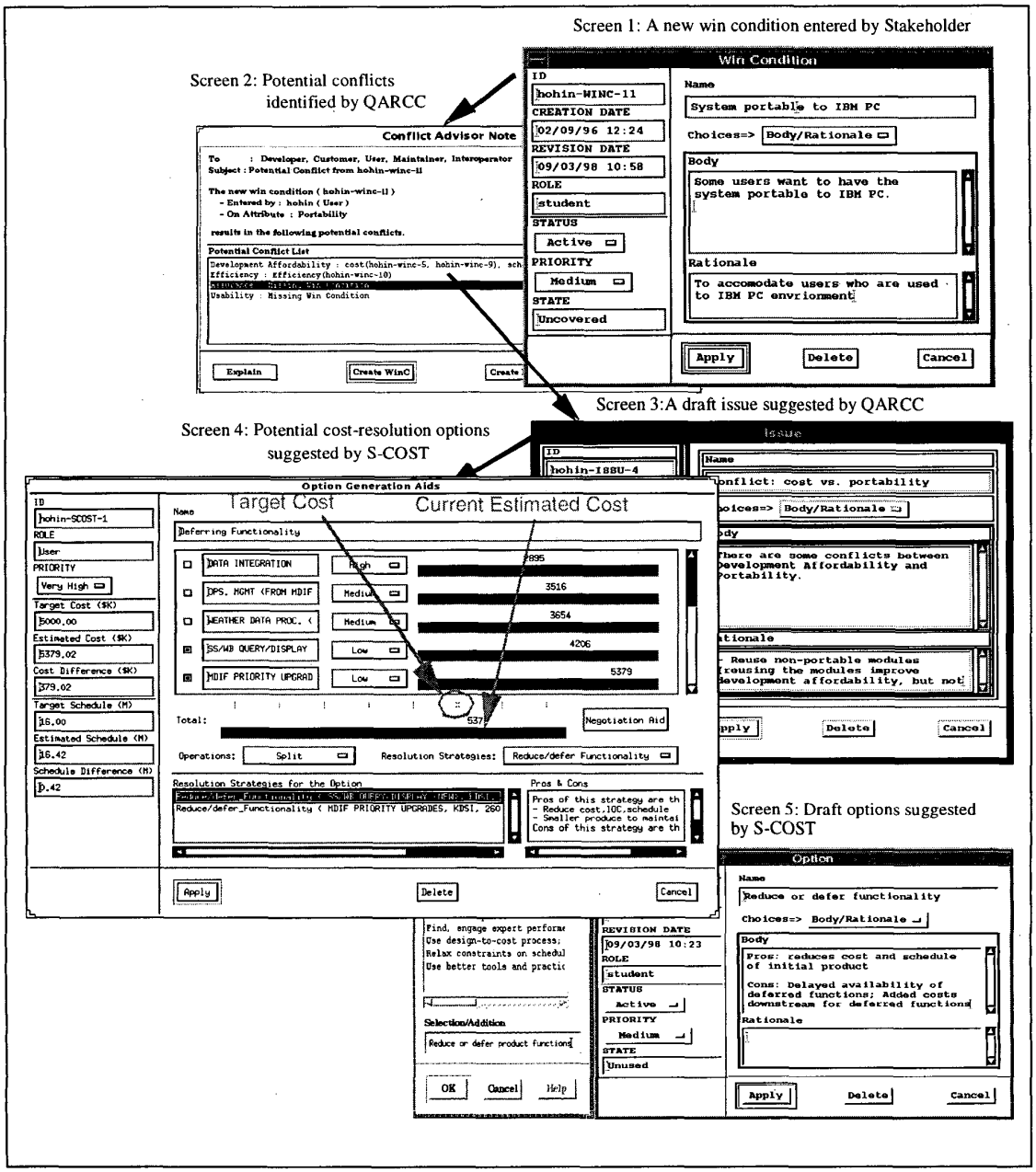
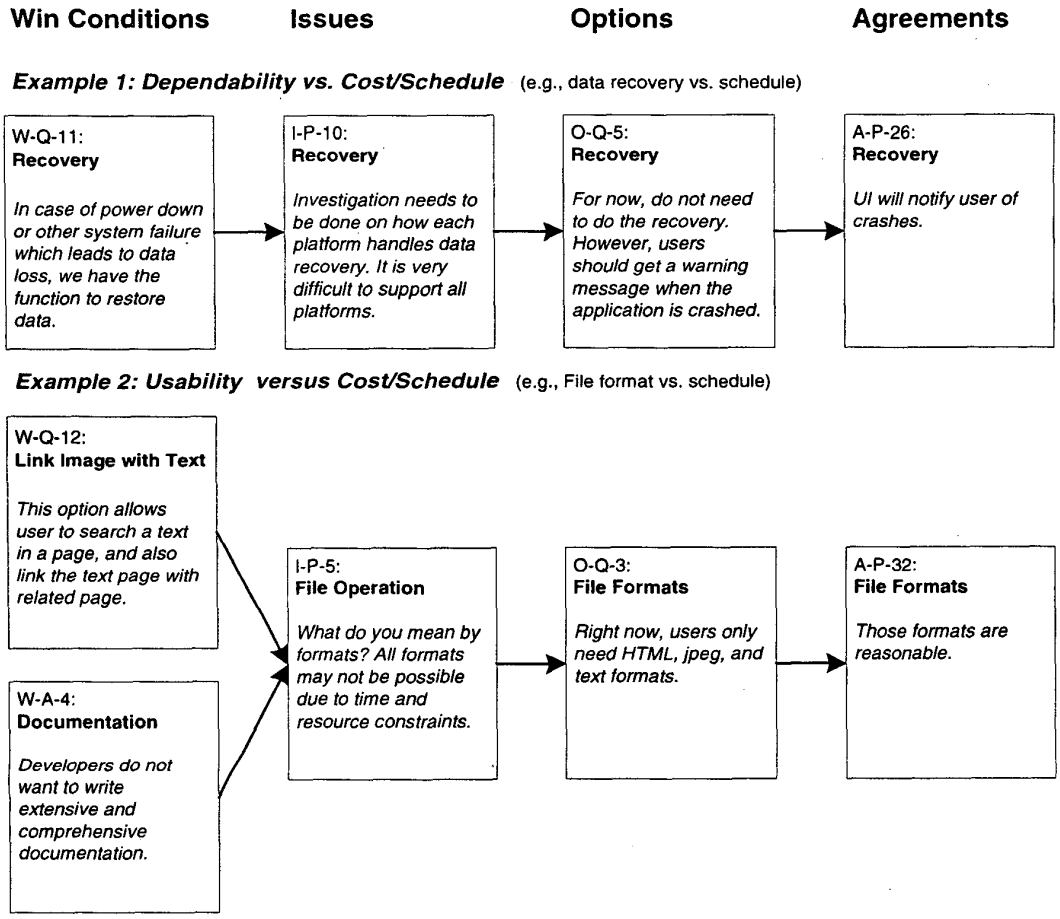



Figure 4 (on the next page) illustrates how Team 2 identified and resolved two quality attribute conflicts. Team 2 used a manual approach (without the aid of QARCC or S-COST). Example 1 resulted in a simple resolution structure (requiring only one win condition, one issue, one option, and one agreement). The entire process focuses on *recovery* of lost data during a power down. A potential conflict occurs between Dependability and Schedule. The issue is resolved with a notification process. Example 2 is slightly more complex. Potential conflicts arise between Usability of a file format and Schedule. Two separate win conditions are linked to a file format issue which is resolved by restricting access to HTML, jpeg and text formatted files. More complex examples would include multiple issues, options and possibly agreements.

**Figure 4: Examples of conflict identification and resolution**




Team 2 identified the seven following quality attribute conflicts using the manual approach.

- **Dependability** versus Cost / Schedule conflicts
  1. *Data Entry*
  2. *Recovery*
- **Usability** versus Cost / Schedule conflicts
  3. *On-line help*
  4. *Concurrent access*
  5. *File format*
- **Reusability** versus Cost / Schedule conflict
  6. *Reusable codes*
- **Performance** versus Cost / Schedule conflict
  7. *Faster Response*

QARCC identified seventy-six potential conflicts. Based on an informed review the following fifty-two conflicts are significant. Twenty-four identified conflicts are determined to be insignificant within the student-based environment (most would be significant within an industrial environment).

**52 Significant quality-attribute conflicts identified by QARCC:**

- **Dependability** versus Cost / Schedule conflicts
  1. *Accuracy optimization*
  2. *Backup/recovery*
  3. *Diagnostics*
  4. *Fault-tolerance functions*
  5. *Integrity functions*
  6. *Interface specification*
  7. *Modularity*
  8. *Monitoring & Control*
  9. *Redundancy*
  10. *UI consistency*
  11. *Undo*
- **Interoperability** versus Cost / Schedule conflicts
  12. *Change source hiding*
  13. *Generality*
  14. *Interface specification*
  15. *Layering*
  16. *Modularity*
  17. *Parameterization*
  18. *Portability*
  19. *UI consistency*
  20. *Understandability*
- **Usability** versus Cost / Schedule conflicts
  21. *Descoping*
  22. *Help/explanation*
  23. *Navigation*
  24. *UI consistency*
  25. *UI flexibility*
  26. *Undo*
  26. *User-programmability*
  27. *User-tailorability*

- **Reusability** versus Cost / Schedule conflict
  *28. Change-source Hiding* *31. Parameterization*
  *29. Interface specification* *32. UI consistency*
  *30. Layering* *33. Understandability*
- **Adaptability** versus Cost / Schedule conflict
  *34. Change-source Hiding* *39. Parameterization*
  *35. Generality* *40. Portability*
  *36. Interface Specification* *41. Self-containedness*
  *37. Layering* *42. UI consistency*
  *38. Modularity*
- **Dependability** versus Performance conflict
  *43. Backup / Recovery* *44. Fault-tolerance*
- **Interoperability** versus Performance conflict
  *45. Portability*
- **Reusability** versus Performance conflict
  *46. Layering*
- **Adaptability** versus Performance conflict
  *47. Layering* *49. Optimization*
  *48. Portability*
- **Cost / Schedule** versus Performance conflict
  *50. Buy faster hardware* *52. Monitoring & Control*
  *51. Optimization*

QARCC handled the seven conflicts identified manually by Team 2, as follows.

1. QARCC **identified** the following 3 conflicts among the seven conflicts that students identified in the WinWin exercise:
   - Data Entry (Error-reducing user input/output, Input acceptability checking, Input assertion/type checking)
   - Recovery (Fault-tolerance, Redundancy, Backup/ Recovery)
   - On-line help (Help/explanation)
2. QARCC identified the following 3 **similar** conflicts among the above conflicts that students identified in the WinWin exercise:
   - Reusable code (can be implemented by Interface specification; Parameterization, Generality, Modulaity)
   - Faster response (can be implemented by Buy faster hardware; monitoring and control, optimization; and platform-feature exploitation)
3. QARCC **could not identify** the following 2 conflicts among the above conflicts that students identified in the WinWin exercise:
   - Concurrent access by multiple users
   - Support multiple file formats

QARCC could not identify domain-specific conflicts (e.g., concurrent access; file format) shown in Category 3. Sometimes, QARCC provided more specific design methods (or strategy) than students did as shown in Category 2.

Based on matching what students drafted as Options, S-COST identifies additional option resolution strategies. Team 2 primarily considered descoping of functionality or quality as the only resolution. For only one option, QARCC did not identify the Team recommended resolution. Following is a comparison of whether S-COST would identify the team recommended resolution.

**Dependability** versus Cost / Schedule conflicts:

*1. Data Entry(Given the time constraint of the semester schedule, it would not be possible to scan all documents.)*
- *Option: The librarians will provide the scanned in images in electronic file.*
- *S-COST Category: Not in S-COST; it is "Out-Sourcing" (need to add this into S-COST KB)*

*2. Recovery(Investigation needs to be done on how each platform handles data recovery. It is very difficult to do it given the time constraint)*
- *Option: For now, we do not need to do the recovery capability. However, users should get a warning message when the application is crashed.*
- *S-COST Category: Descoping_functionality*

**Usability** versus Cost / Schedule conflicts:

*3. On-line help: Schedule is tight. We do not have time to write a lot of document.*
- *Option: Since functionality is not very complex, how about writing simple on-line help?*
- *S-COST Category: Descoping_functionality*

*4. Concurrent access: The number of concurrent users depends on the power of the server the customer agrees to purchase.*
- *Option: USC will supply the hardware. Since this is a $0 budget, whatever they have will be supplied.*
- *S-COST Category: Descoping_functionality*

*5. File format: What do you mean by the formats? Not all formats may be possible due to time and resource constraints.*
- *Option: Right now, users only need HTML, jpeg and text formats.*
- *S-COST Category: Descoping_functionality*

**Reusability** versus Cost / Schedule conflict:

*6. Reusable codes: In order to cut down future cost, immediate cost may be necessary if the best option in terms of reusability is agreed upon.*
- *Option: Since time is the utmost concern, we need to finish the project by the end of this semester. Forget about reusability*
- *S-COST Category: Descoping_quality*

**Performance** versus Cost / Schedule conflict:

*7. Faster Response: Customer must be willing to obtain hardware / server to support these requirements. A faster processor is needed.*
- *Option: Please try to do the best for the performance within the given time.*
- *S-COST Category: Descoping_quality*

Why did students not consider other viable option strategies? Possibly, students only considered reducing or deferring functionality given requirements for a high

degree of functionality combined with a tight class schedule and a small supporting budget. Possibly, librarians could not provide more options (e.g., buy new tools or reuse COTS components) due to lack of digital library project funding.

On the other hand, discussion between the students and the librarians may not be recorded into WinWin due to input barriers (e.g., availability of computer facilities during the discussion, unwillingness of the librarians to use the computers), although the conclusion usually is. S-COST can capture most options that students drafted (six out of seven). S-COST also can help students consider eleven viable option strategies (and/or a combination of these):

1. Reduce/defer functionality
2. Reduce/defer software quality
3. Improve tools, techniques or platforms
4. Relax the delivery schedule constraint
5. Improve personnel capabilities
6. Re-use software assets
7. Improve coordination of multiple project stakeholders as a team
8. Architecture and risk resolution
9. Improve process maturity level
10. Improve precedentedness and development flexibility
11. Increase budget

Not all eleven strategies are applicable or significant in every situation. For example, *"Relax the delivery schedule constraint"* may be not be viable due to the class schedule constraint (i.e., the class must finish by the end of the semester). Five strategies (i.e., strategy 1, 2, 3, 5 and 6) were considered to be significant ones during this S-COST case study given the availability of freeware and commercial tools, training through group study, and reusable software components developed by other classes. Note that students used only three strategies (i.e., strategy 1, 2, and out-sourcing which is not in the S-COST knowledge base).

## 4. EMPIRICAL FINDINGS

Team 2 is representative of the other fourteen teams. For most teams, significant conflicts resulted from Cost and Schedule related issues. Figure 5 summarizes the mapping of stakeholder interests anticipated by the WinWin Model with the actual mapping resulting from the fifteen teams. It is interesting, and possibly not surprising, that the Customer (librarian staff) was significantly more interested in unanticipated quality attributes (Dependability, Interoperability, and Usability) than the the student developers.

Table 1 (on the next page) summarizes research hypotheses concerning stakeholder behaviors within the negotiation process, analyses, and implications. Four basic findings were observed:

1. **Stakeholders accept satisfactory rather than optimal solutions.** Most issues are non-controversial and easy to resolve. Stakeholders tend to directly accept options as presented or merge options into an acceptable solution.
2. **Stakeholder interests can not necessarily be anticipated.** Accommodation must be made for specific situations.
3. **Users and Customers are more proactive in stating win conditions while developers are more active in working toward resolution.** While intuitively appealing, the implication is that developers should be better trained to anticipate win conditions.
4. **Knowledge-based automated aids provide improvements in conflict identification and resolution.** QARCC and S-COST identify most issues and resolution strategies considered by student teams. In addition, QARCC and S-COST identify a significant number of additional issues and solution strategies than considered by manual approaches.

## 5. FUTURE DIRECTIONS

Student-based digital library projects continued throughout the remainder of the 1990s. Refinements and advances have been made to the WinWin process and supporting tools. Additional universities have also adopted similar processes and tool support. During 2000, the WinWin process was slightly modified and incorporated into electronic meeting system technology. The adoption is referred to as EasyWinWin and is based on GroupSystems.com technology. Related research and tool development continues within academic, government and commercial environments.

From a research perspective, issues currently being examined include:

- Is the process of conflict resolution for quality-attributes similar to a general conflict issue resolution process? What are specific patterns of quality-conflict resolution?
- Which stakeholders are primarily concerned with which quality attributes?
- How can expertise be captured and made broadly available via automated aids for quality-conflict resolution?

*Figure 5: Anticipated and Actual Mapping of Stakeholder Interests in Quality Attributes*

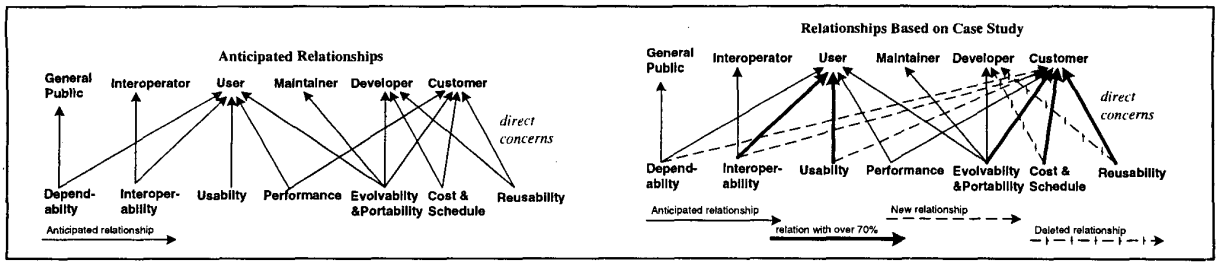



**Table 1. Summary of Empirical Findings**

| HYPOTHESIS | ANALYSIS/FINDINGS | IMPLICATION |
|---|---|---|
| 1. Most quality-attribute win conditions will be non-controversial. | • 55% of goals are non-controversial | • A tendency exists to gravitate to satisfactory solutions and not necessarily optimal solutions unless the issue is at the core of the stakeholder interest. |
| 2. Most quality issues will be simple to resolve. | • 66% of issues have one option<br>• 15% of issues have two options | |
| 3. Among the conflict issues having only one option, the number of issues resolved by *directly accepting* options will be greater than those resolved by *adapting* options. | • Supported but not statistically significant<br>• *Decomposing* and *Contradictorily Accepting* patterns are rare (3%) | |
| 4. Among the conflict issues having multiple options, the number of issues resolved by *electing the best option(s)* will be greater than the number of issues resolved by *merging multiple options* or *accepting all the multiple options*. | • 42% Electing the best option<br>• 26% Merging multiple options<br>• 32% Accepting all multiple options | |
| 5. The primary relationship between stakeholders and quality attributes shown in Figure 3 will hold for the digital library student projects. | • Primary relationship holds; however unanticipated relationships are also created. | • Must analyze interests of individual stakeholders and accommodate for specific situations. |
| 6. **Users** will be more active in identifying win conditions, but less active in identifying issues, options, and agreements. | • 55% Win Conditions<br>• 10% Issues, 17% Options, and 18% Agreements | • Users and customers have direct interests that affect their jobs because they are more active in identifying win conditions (their interests) but their inactivity in developing resolutions transfers the burden to developers. Generally speaking developers are not well trained in conflict resolution and negotiation in order to resolve win conditions. |
| 7. **Customers** will be more active in identifying win conditions, but less active in identifying issues, options, and agreements. | • 46% Win Conditions<br>• 13% Issues, 18% Options, and 23% Agreements | |
| 8. **Developers** will be more active in identifying issues and agreements, but less active in identifying win conditions and options. | • 33% Issues and 38% Agreements<br>• 11% Win Conditions and 18% Options, | |
| 9. QARCC will identify a significant number of issues raised by the manual approach. | • 79% of issues identified by QARCC<br>• 15% of issues not identified by QARCC | • Semi-automatic knowledge-based tools significantly cover issues and options identified by manual approaches.<br>• Further, such tools greatly expand the number of issues and options considered by manual approaches. |
| 10. The number of significant quality-conflict issues raised by QARCC is much greater than the number of conflict issues raised by students using the manual approach. | • QARCC surfaced 742% (52 vs. 7) more issues than generated by the manual approach | |
| 11. S-COST will identify a significant number of conflict resolution options generated by the manual approach | • 97% strategies identified by S-COST | |
| 12. The number of significant cost-conflict resolution options in the S-COST knowledge base is much greater than the number of cost conflict-resolution options by students using the manual approach. | • S-COST surfaced 166 % (5 vs. 3) more solution strategies than generated by the manual approach | |

From a tool building and process perspective, issues currently being examined include:

- How can the WinWin taxonomy be customized to the specific situation? The ability to add attributes is needed.
- Is process enforcement appropriate? Agreements sometimes occur without drafting options (information loss occurs).
- How can conflicts be easily classified? Many conflicts partially affect two or more quality attributes and better methods are needed to handle these situations.
- How do team factors (industry-experienced students vs. non-experienced students) affect the ability to identify and resolve quality-attribute conflicts?

Thus far the research and tool building efforts have contributed to better practice. Specifically, we have a better grasp of tradeoffs and interactions among quality requirements. Further, case studies provide examples of how quality requirement conflicts can be identified and resolved.

## 6. RELATED WORK

Early characterizations of software quality attribute relationships based on hierarchical models were developed in [10] and [11]. The hierarchical models of software quality attributes are based upon a set of quality criteria, each of which has a set of measures or metrics associated with it. Rome Laboratory sponsored a number of follow-on projects to the McCall study, including [12-14]

More recent work focuses on quality analysis methods in software architecture, one of the most important tools for designing and understanding a software system. The manual scenario-based Software Attribute Analysis Method (SAAM) [15] provides a technique for analyzing the architecture under consideration with respect to how well or easily it satisfies the constraints imposed by each scenario. The scenarios were used to express the particular instances of each quality attribute important to the customer of a system. One of their difficulties is determining the proper set of the scenarios.

Further, CMU-SEI work on software quality attributes [16] explored the relationship between the software architecture of a system and the software qualities to be achieved by the system. The relationship is based on design operations (called "unit operations") such as separation, abstraction, compression, composition, resource sharing, and replication. Their work provides a useful first-order conflict analysis of the interaction among quality attributes, though their method of deriving architectures from requirements is somewhat oversimplified. Recently, they have begun to focus on elaboration of the interaction in Architecture Tradeoff Analysis Method (ATAM) [17] and Attribute-Based Architecture Styles (ABASs) [18]. Our research focus in QARCC and S-COST [4, 6, 9] used this concept and extended it with a semi-formalized structure in order to provide a useful description language for architecture tradeoff analysis.

University of Toronto researchers [19] developed automated assistance in dealing with interactions among non-functional quality requirements based on Non-Functional Requirements (NFRs). They focused on traceability of quality requirements with more emphasis on incorporating changes in NFRs (e.g., systematically detecting defects and supporting the process of corresponding changes in design and implementation). However, there is no consideration of stakeholders who have different quality-attribute priorities and concerns.

## 7. CONCLUSION

This case study has furnished an experience factory to follow stakeholder behavior patterns in resolving system quality attributes. Quality attributes are strong drivers, usually coequal with functional scope, on system cost and schedule. Frequently these matters are left unexplored in the project with differing expectations by the project's stakeholders. The major findings of this study with respect to the two research questions raised in the Introduction are:

1. *Behavioral differences between stakeholders:*

- Stakeholders usually accept reasonable solutions and escalate by exception only those conflict issues central to their individual interests;
- Users and customers are more active than developers when identifying win conditions. The active role shifts to the developer in the resolution portion of the cycle.

2. *Enhancements provided by automated aids:*

- QARCC and S-COST collectively surface a larger number of quality attribute issues and options than those raised manually by the stakeholders;
- Some of the QARCC/S-COST originated issues were false alarms. It was relatively trivial to filter out the specious items. This favors the potential of these automated aids to substantially augment the basic WinWin model as applied to quality attributes.
- If QARCC is trained as to the nature of the system under consideration (e.g., mission-critical, academic information system, commercial information system, real-time systems), then "false alarms" can be greatly reduced by turning off secondary-concerned qualities.

QARCC does not assess conflict significance. Nonetheless, it is a valuable (cost-effective) knowledge based tool that provides a check-list for conflict identification.

It is always tenuous to extrapolate case study findings to the general practice of software engineering. There is a severe shortage of good experienced system architects under whose domain quality attribute determination and

tradeoffs would normally fall. It is here that we cautiously extend the experience of this case study to note that these knowledge-based aids can assist inexperienced architects and backfill this shortage.


## ACKNOWLEDGEMENTS

This work is partially supported by funding from NASA JPL under the contract C00-00443 with Texas A&M University. In addition, this research wais partially sponsored by the Advanced Research Projects Agency (ARPA) through Rome Laboratory under contract F30602-94-C-0195 to USC and by the Affiliates of the USC Center for Software Engineering: Allied Signal Corp., Bellcore, Boeing, Electronic Data Systems Corporation, E-Systems, FAA, GDE systems, Hughes Aircraft Company, Interactive Development Environments, Institute for Defense Analysis, Jet Propulsion Laboratory, Litton Data Systems, Lockheed Martin Corporation, MCC, Motorola Inc., Northrop Grumman Corporation, Rational Software Corporation, Raytheon, Science Applications International Corporation, Software Engineering Institute (CMU), Software Productivity Consortium, Sun Microsystems, Inc., Texas Instruments, TRW, U.S. Air Force Rome Laboratory, U.S. Army Research Laboratory, and Xerox Corporation.


## REFERENCES


1. Boehm, B., Bose, P., Horowitz, E., and Lee, M., *"Software Requirements As Negotiated Win Conditions", First International Conference on Requirements Engineering (ICRE94).* 1994. Colorado Springs, Colorado: IEEE Computer Society Press.
2. Boehm, B., Bose, P., Horowitz, E., and Lee, M., *"Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach", 17th International Conference on Software Engineering (ICSE-17).* 1995. Seattle, WA: IEEE Computer Society Press.
3. Boehm, B., Bose, P., Horowitz, E., and Lee, M., *"Cost Models for Future Life-Cycle Processes: COCOMO 2.0",* Annals of Software Engineering, 1995: p. 57-94.
4. Boehm, B. and H. In, *"Identifying Quality-Requirement Conflicts",* IEEE Software, 1996. March: p. 25-35.
5. Boehm, B. and H. In, *"Cost vs. Quality Requirements: Conflict Analysis and Negotiation Aids",* Software Quality Professional, 1999. 1(2): p. 38-50.
6. Boehm, B. and H. In, *"Software Cost Option Strategy Tool (S-COST)", COMPSAC96 (The Twentieth Annual International Computer Software and Applications Conference).* 1996. Seoul, Korea: IEEE Comp. Society Press.
7. Boehm, B. and R. Ross, *"Theory W Software Project Management: Principles and Examples",* IEEE Transactions on Software Engineering, 1989. July: p. 902-916.
8. Egyed, A. and B. Boehm, *"Analysis of Software Requirements Negotiation Behavior Patterns", International Conferences of Systems Engineering (INCOSE).* 1997.
9. Boehm, B., *Egyed, A., Port, D., Shah, A., Kwan, J., Madachy, R., "A Stakeholder Win-Win Approach to Software Engineering Education",* Annals of Software Engineering, 1999.
10. Boehm, B., J. Brown, and M. Lipow, *"Quantitative Evaluation of Software Quality", Second International Conference on Software Engineering (ICSE-76).* 1976.
11. McCall, J.A., P.K. Richards, and G.F. Walters, *"Factors in Software Quality",* 1977, Rome Laboratory: Griffiss Air Force Base, NY.
12. Bowen, T., G. Wigle, and J. Tsai, *"Specifications of Quality Attributes",* 1985, Rome Air Development Center: Griffiss Air Force Base, NY,. p. Vol. I-III.
13. Lasky, J. and D. K., *"Conflict Resolution (CORE) for Software Quality Factors",* 1993, Rome Lab: Griffis Air Force Base, NY.
14. Murine, G. and B. Murine, *"Implementing a Software Quality Metric Program Based on the Rome Laboratory Initiatives",* Annals of Software Engineering, 1995. 1: p. 155-177.
15. Kazman, R. and L. Bass, *"Toward Deriving Software Architectures From Quality Attributes",* 1994, SEI/CMU: Pittsburgh, PA.
16. Kazman, R., *Bass, L., Abowd, G., and Webb, M., "SAAM: A Method for Analyzing the Properties of Software Architectures", 16th Conference of Software Engineering (ICSE-16).* 1994. Sorrento, Italy: IEEE Computer Society Press.
17. Kazman, R., Barbacci, M., Klein, M., Carriere, S.J., *"Experience with Performing Architecture Tradeoff Analysis", International Conference of Software Engineering (ICSE-99).* 1999. Los Angeles, CA.
18. Klein, M., *Kazman, R., Bass, L., Carriere, S.J., Barbacci, M., Lipson, H., "Attribute-Based Architectural Styles", First Working IFIP Conference on Software Architecture (WICSA1).* 1999. San Antonio, TX.
19. Chung, L., B. Nixon, and E. Yu, *"Using Non-Functional Requirements to Systematically Support Change", Second International Conference of Requirements Engineering (ICRE-2).* 1995: IEEE Computer Press.